IN THE SPECIFICATION

Please replace the paragraph beginning at line 21, page 1 and ending at line 2, page 2 with the following rewritten paragraph:

Internationalization is a process of enabling a program, e.g., [[Java]] Java™, to run internationally. That is, an internationalized program has the flexibility to run correctly in any country. An internationalized program must be able to read, write and manipulate localized text. Furthermore, an internationalized program must conform to local customs when displaying dates and times, formatting numbers and sorting strings.

a1

Please replace the paragraph beginning at line 3, page 2 with the following rewritten paragraph:

Internationalization is becoming increasingly important with the explosive growth of the Internet and the World Wide Web where an ever increasing number of computer users are from various locales. A locale represents a geographic, cultural or political region. One of the problems with internationalization involves the use of text strings that may be hard-coded in the program, e.g., [[Java]] Java™. Hard-coded text strings refer to text that will not vary with the locale. That is, the text strings may appear in English even when the program is run on the French locale. Various object-oriented languages such as [[Java]] Java™ have developed tools to assist in developing internationalized programs and allowing text strings to appear in the language of the locale. A discussion of object-oriented programming languages and in particular Java is deemed appropriate.

a2

Please replace the paragraph beginning at line 13, page 2 with the following rewritten paragraph:

In an object-oriented programming language such as [[Java]] Java™, a class is a collection of data and methods that operate on that data. The data and methods taken together describe the state and behavior of what is commonly referred to as an object. An object in essence includes data and code where the code manipulates the

a3

a3

data. Hence a software application may be written using an object-oriented programming language such as [[Java]] Java™ whereby the program's functionality is implemented using objects.

Please replace the paragraph beginning at line 19, page 2 with the following rewritten paragraph:

a4

Unlike many programming languages, [[Java]] Java™ is compiled into machine independent code commonly referred to as bytecodes instead of machine dependent code, i.e. executable code. Bytecodes are stored in a particular file format commonly referred to as a "class file" that includes bytecodes for methods of a class. In addition to the bytecodes for methods of a class, the class file includes a symbol file as well as other ancillary information.

Please replace the paragraph beginning at line 1, page 3 with the following rewritten paragraph:

a5

A computer program embodied as [[Java]] Java™ bytecodes in one or more class files is platform independent. The computer program may be executed, unmodified, on any computer that is able to run an implementation of what is commonly referred to as a [[Java]] Java™ virtual machine. The [[Java]] Java™ virtual machine is not an actual hardware platform, but rather a low level software emulator that can be implemented on many different computer processor architectures and under many different operating systems. The [[Java]] Java™ virtual machine reads and interprets each bytecode so that the instructions may be executed by the native processor. Hence a [[Java]] Java™ bytecode is capable of functioning on any platform that has a [[Java]] Java™ virtual machine implementation available. However, bytecode interpretation detracts from processor performance since the microprocessor has to spend some of its processing time interpreting bytecode instructions. Compilers commonly referred to as "just in time (JIT)" were developed to improve the performance of Java virtual machines. A JIT compiler translates [[Java]] Java™ bytecodes into the processor's native machine code during runtime. The processor then executes the compiled native machine code.

Please replace the paragraph beginning at line 16, page 3 and ending at line 3, page 4 with the following rewritten paragraph:

As stated above [[Java]] Java™ has developed tools to assist in developing internationalized programs and allowing text strings to appear in the language of the locale. One such tool is the use of resource files commonly referred to in [[Java]] Java™ as resource bundles. A resource bundle class may be used for externalizing text strings, i.e. not hard-coding strings in the program. The resource bundle class represents a bundle of resources that may be looked up by name. The resources may include appropriate text strings for a given locale that are indexed by what are commonly referred to as keys. Keys are free formatted strings that appear in the program code as well as in the resource bundle thereby allowing the program to access the externalized string. By having resource bundles associated with particular locales, e.g., a resource file with resources associated with the US English locale, a resource file with resources associated with the French locale and so forth, appropriate text strings associated with the particular locale may be loaded at runtime.

Please replace the paragraph beginning at line 4, page 4 with the following rewritten paragraph:

However, software developers may still hard-code their strings into their application instead of externalizing them and loading them from the resource bundle. Various scanning programs have been developed which attempt to detect hard-coded strings. Unfortunately, these scanning programs simply detect as hard-coded strings all text enclosed within double quotes ("") which are used as string delimiters in [[Java]] Java™ (as well as other programming languages). However, not all text enclosed within double quotes are hard-coded strings. The text enclosed within the double quotes may be a path name to a resource file, e.g., resource bundle.

Please replace the paragraph beginning at line 3, page 5 with the following rewritten paragraph:

The problems outlined above may at least in part be solved in some embodiments by a scanning program that scans a code, e.g., [[Java]] Java™, line by line until a pair of string delimiters is identified. Once a pair of string delimiters is identified, the scanning program determines whether the string within the pair of string delimiters identified is a path name to a resource file, e.g., resource bundle. If the string is a path name to the resource file, then the string is not a hard-coded string. If the string is not a path name to the resource file, then the string may be identified as a possible hard-coded string.

a8

Please replace the paragraph beginning at line 3, page 7 with the following rewritten paragraph:

The present invention comprises a method, computer program product and data processing system for identifying non-externalized strings that are not hard-coded. In one embodiment of the present invention a scanning program scans a code, e.g., [[Java]] Java™, line by line until a pair of string delimiters is identified. Once a pair of string delimiters is identified, the scanning program determines whether the string within the pair of string delimiters identified is a path name to a resource file, e.g., resource bundle. If the string is a path name to the resource file, then the string is a non-externalized string that is not hard-coded. If the string is not a path name to the resource file, then the string may be identified as a possible hard-coded string. It is noted that the even though the following discusses the present invention in conjunction with a [[Java]] Java™ programming environment the present invention may be implemented in any type of programming environment where the programming language has the capability of externalizing text strings in resource files.

a9

Please replace the paragraph beginning at line 19, page 7 and ending at line 20, page 8 with the following rewritten paragraph:

Figure 1 illustrates a typical hardware configuration of data processing system 13 which is representative of a hardware environment for practicing the present invention. Data processing system 13 has a central processing unit (CPU) 10, such as

a10

a conventional microprocessor, coupled to various other components by system bus 12. An operating system 40, e.g., DOS, OS/2™, runs on CPU 10 and provides control and coordinates the function of the various components of Figure 1. An object-oriented programming system, such as [[Java]] Java™ 42, runs in conjunction with operating system 40 and provides output calls to operating system 40 which implements the various functions to be performed by the application 42. Read only memory (ROM) 16 is coupled to system bus 12 and includes a basic input/output system ("BIOS") that controls certain basic functions of data processing system 13. Random access memory (RAM) 14, I/O adapter 18, and communications adapter 34 are also coupled to system bus 12. It should be noted that software components including operating system 40 and application 42 are loaded into RAM 14 which is the computer system's main memory. I/O adapter 18 may be a small computer system interface ("SCSI") adapter that communicates with disk units 20, e.g., disk drive, and tape drives 40. It is noted that the scanning program of the present invention that identifies non-externalized strings that are not hard-coded may reside in disk unit 20 or in application 42. Communications adapter 34 interconnects bus 12 with an outside network enabling data processing system 13 to communication with other such systems. Input/Output devices are also connected to system bus 12 via a user interface adapter 22 and a display adapter 36. Keyboard 24, trackball 28, mouse 26 and speaker 30 are all interconnected to bus 12 through user interface adapter 22. Event data may be input to the object-oriented programming system through any of these devices. A display monitor 38 is connected to system bus 12 by display adapter 36. In this manner, a user is capable of inputting to system 13 through keyboard 24, trackball 28 or mouse 26 and receiving output from system 13 via display 38 or speaker 30.

Please replace the paragraph beginning at line 15, page 9 and ending at line 2, page 10 with the following rewritten paragraph:

Figure 2 illustrates a method 200 for identifying non-externalized strings that are not hard-coded. As stated in the Background Information section, software developers may hard-code their strings into their application, e.g., [[Java]] Java™,

instead of externalizing them and loading them from the external resource file, e.g., resource bundle. Various scanning programs have been developed which attempt to detect hard-coded strings. Unfortunately, these scanning programs simply detect as hard-coded strings all text enclosed within double quotes, i.e. string delimiters. However, not all text enclosed within double quotes are hard-coded strings. The text enclosed within the double quotes may be a path name to a resource file, e.g., resource bundle. Method 200 is a method that identifies non-externalized strings, e.g., path names to resource files, that are not hard-coded that are enclosed within string delimiters.

a11

Please replace the paragraph beginning at line 3, page 10 with the following rewritten paragraph:

In step 210, a scanning program scans the code of an application program 42 line by line for string delimiters until the scanning program identifies a pair of string delimiters. String delimiters refers to the quotes ("") that mark the beginning and end of a text string. For example, in the classic Hello World program written in [[Java]] Java™ as shown below

a12

```
public class HelloWorld  {
  public static void main
  (String args[])  {
    System.out.printin("Hello World");
  }
}
```

the string delimiters mark the beginning and end of the text string "Hello World."

Please replace the paragraph beginning at line 17, page 10 and ending at line 17, page 11 with the following rewritten paragraph:

A determination is then made in step 220 as to whether the scanning program has identified a pair of string delimiters that mark the beginning and end of a text string. If the scanning program has not identified a pair of string delimiters, then

a13

method 200 is terminated at step 230. If the scanning program has identified a pair of string delimiters, the scanning program determines whether the string within the string delimiters is a path name to the resource file in step 240. As stated in the Background Information section, a resource file is commonly referred to as a resource bundle in [[Java]] Java™. It is further noted that a path name to a resource file, e.g., resource bundle, is a non-externalized string that is not hard-coded but is enclosed within double quotes within a program, e.g., [[Java]] Java™. For example, in the [[Java]] Java™ code shown below

```
rbCal=ResourceLoader.getBundle("com.tivoli.uif.Resources.CalendarResources");
setTitle(ResourceLoader.getString(rbCal, "Holiday Title");
```

a13

the string "com.tivoli.uif.Resources.CalendarResources" is a path name to a resource bundle. Path names to resource bundles are commonly referred to as uniform resource locator (URL). URL's are commonly identified by their dotted signature. It is noted that rbCal is a resource bundle where the method ResourceLoader retrieves calender resources from the URL "com.tivoli.uif.Resources.CalendarResources." It is further noted that the second line of the above written [[Java]] Java™ code sets the title to the window to an externalized string "Holiday Title" located in the resource bundle, rbCal. That is, the method ResourceLoader will retrieve the proper string associated with the language of the locale. For example, if the locale is an American locale, then the holiday title on the window may appear as "Christmas." If the locale is a Spanish locale, then the holiday title on the window may appear as "Navidad."

Please replace the paragraph beginning at line 18, page 11 and ending at line 12, page 12 with the following rewritten paragraph:

a14

Referring to the above example of [[Java]] Java™ code, the scanning program may identify the string delimiters that mark the beginning and end of the text string "com.tivoli.uif.Resources.CalendarResources" in step 220. In step 240, a determination is made by the scanning program as to whether the string within the string delimiters identified in step 220 is a non-externalized string that is not

hard-coded. If the scanning program determines that the string within the string delimiter is a non-externalized string that is not hard-coded, then the scanning program will not flag the string as a possible hard-coded string in step 250. As stated above, the scanning program will identify the string within the string delimiters as a non-externalized string that is not hard-coded if the string is a URL identified from its dotted signature, e.g., "com.tivoli.uif.Resources.CalendarResources." A determination is then made in step 260 as to whether there is any more code to scan by the scanning program. If there is no more code to scan by the scanning program, then method 200 is terminated at step 230. If there is more code to scan by the scanning program, then method 200 proceeds to scan the remaining code for string delimiters in step 210. It is noted for clarity that if the scanning program has identified a pair of string delimiters in a particular line of code in step 220 and there is more code within that particular line, then the scanning program may continue to scan the remainder of that particular line of code and the remaining line(s) of code until the scanning program identifies a pair of string delimiters.

a14